LAMPS
Internet-Draft
Intended status: Standards Track
Expires: 5 September 2024

M. Ounsworth
J. Gray
Entrust
M. Pala
CableLabs
J. Klaussner
D-Trust GmbH

4 March 2024

Composite ML-DSA for use in Internet PKI
draft-ounsworth-pq-composite-sigs-13

Abstract

This document defines Post-Quantum / Traditional composite Key Signaturem algorithms suitable for use within X.509, PKIX and CMS protocols. Composite algorithms are provided which combine ML-DSA with RSA, ECDSA, Ed25519, and Ed448. The provided set of composite algorithms should meet most X.509, PKIX, and CMS needs.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 September 2024.

Copyright Notice

Table of Contents

1. Changes in version -13

* Shortened Abstract.

* Added text to Introduction to justify where and why this mechanism would be used.

* Resolved comments from Kris Kwiatkowski

* Resolved Key Usage comments from Tim Hollebeek

* Fixed up Algorithm names in Algorithm Deprecation section

* Removed Falcon composites to not delay the release of this document. Falcon (FN-DSA) composites can be added in a separate document

* Add a security consideration about Trust Anchors

* Updated the included samples to conform to this draft

2. Introduction

During the transition to post-quantum cryptography, there will be uncertainty as to the strength of cryptographic algorithms; we will no longer fully trust traditional cryptography such as RSA, Diffie-Hellman, DSA and their elliptic curve variants, but we will also not fully trust their post-quantum replacements until they have had sufficient scrutiny and time to discover and fix implementation bugs. Unlike previous cryptographic algorithm migrations, the choice of when to migrate and which algorithms to migrate to, is not so clear. Even after the migration period, it may be advantageous for an entity's cryptographic identity to be composed of multiple public-key algorithms.

Cautious implementers may wish to combine cryptographic algorithms such that an attacker would need to break all of them in order to compromise the data being protected. Such mechanisms are referred to as Post-Quantum / Traditional Hybrids [I-D.driscoll-pqt-hybrid-terminology].

In particular, certain jurisdictions are recommending or requiring that PQC lattice schemes only be used within a PQ/T hybrid. As an example, we point to [BSI2021] which includes the following recommendation:

"Therefore, quantum computer-resistant methods should not be used alone - at least in a transitional period - but only in hybrid mode, i.e. in combination with a classical method. For this purpose, protocols must be modified or supplemented accordingly. In addition, public key infrastructures, for example, must also be adapted"

This specification represents the straightforward implementation of the hybrid solutions called for by European cyber security agencies.

PQ/T Hybrid cryptography can, in general, provide solutions to two migration problems:

* Algorithm strength uncertainty: During the transition period, some post-quantum signature and encryption algorithms will not be fully trusted, while also the trust in legacy public key algorithms will start to erode. A relying party may learn some time after deployment that a public key algorithm has become untrustworthy, but in the interim, they may not know which algorithm an adversary has compromised.

* Ease-of-migration: During the transition period, systems will require mechanisms that allow for staged migrations from fully classical to fully post-quantum-aware cryptography.

* Safeguard against faulty algorithm implementations and compromised keys: Even for long known algorithms there is a non-negligible risk of severe implementation faults. Latest examples are the ROCA attack and ECDSA psychic signatures. Using more than one algorithms will mitigate these risks.

This document defines a specific instantiation of the PQ/T Hybrid paradigm called "composite" where multiple cryptographic algorithms are combined to form a single signature such that it can be treated as a single atomic algorithm at the protocol level. Composite algorithms address algorithm strength uncertainty because the composite algorithm remains strong so long as one of its components remains strong. Concrete instantiations of composite signature algorithms are provided based on ML-DSA, RSA and ECDSA. Backwards compatibility is not directly covered in this document, but is the subject of Appendix B.2.

This document is intended for general applicability anywhere that digital signatures are used within PKIX and CMS structures. For a more detailed use-case discussion for composite signatures, the reader is encouraged to look at [I-D.vaira-pquip-pqc-use-cases]

This document attemps to bind the composite component keys together to achieve the weak non-separability property as defined in [I-D.hale-pquip-hybrid-signature-spectrums] using a label as defined in [Bindel2017].

2.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used in this document:

ALGORITHM: A standardized cryptographic primitive, as well as any ASN.1 structures needed for encoding data and metadata needed to use the algorithm. This document is primarily concerned with algorithms for producing digital signatures.

BER: Basic Encoding Rules (BER) as defined in [X.690].

CLIENT: Any software that is making use of a cryptographic key. This includes a signer, verifier, encrypter, decrypter.

COMPONENT ALGORITHM: A single basic algorithm which is contained within a composite algorithm.

COMPOSITE ALGORITHM: An algorithm which is a sequence of two component algorithms, as defined in Section 3.

DER: Distinguished Encoding Rules as defined in [X.690].

LEGACY: For the purposes of this document, a legacy algorithm is any cryptographic algorithm currently in use which is not believed to be resistant to quantum cryptanalysis.

PKI: Public Key Infrastructure, as defined in [RFC5280].

POST-QUANTUM ALGORITHM: Any cryptographic algorithm which is believed to be resistant to classical and quantum cryptanalysis, such as the algorithms being considered for standardization by NIST.

PUBLIC / PRIVATE KEY: The public and private portion of an asymmetric cryptographic key, making no assumptions about which algorithm.

SIGNATURE: A digital cryptographic signature, making no assumptions about which algorithm.

STRIPPING ATTACK: An attack in which the attacker is able to downgrade the cryptographic object to an attacker-chosen subset of original set of component algorithms in such a way that it is not detectable by the receiver. For example, substituting a composite public key or signature for a version with fewer components.

2.2. Composite Design Philosophy

[I-D.driscoll-pqt-hybrid-terminology] defines composites as:

> _Composite Cryptographic Element_: A cryptographic element that incorporates multiple component cryptographic elements of the same type in a multi-algorithm scheme.

Composite keys as defined here follow this definition and should be regarded as a single key that performs a single cryptographic operation such key generation, signing, verifying, encapsulating, or decapsulating -- using its internal sequence of component keys as if they form a single key. This generally means that the complexity of combining algorithms can and should be handled by the cryptographic library or cryptographic module, and the single composite public key, private key, and ciphertext can be carried in existing fields in protocols such as PKCS#10 [RFC2986], CMP [RFC4210], X.509 [RFC5280], CMS [RFC5652], and the Trust Anchor Format [RFC5914]. In this way, composites achieve "protocol backwards-compatibility" in that they will drop cleanly into any protocol that accepts signature algorithms without requiring any modification of the protocol to handle multiple keys.

2.3. Composite Signatures

Here we define the signature mechanism in which a signature is a cryptographic primitive that consists of three algorithms:

* KeyGen() -> (pk, sk): A probabilistic key generation algorithm, which generates a public key pk and a secret key sk.

* Sign(sk, Message) -> (signature): A signing algorithm which takes as input a secret key sk and a Message, and outputs a signature

* Verify(pk, Message, signature) -> true or false: A verification algorithm which takes as input a public key, a Message and signature and outputs true if the signature verifies correctly. Thus it proves the Message was signed with the secret key associated with the public key and verifies the integrity of the Message. If the signature and public key cannot verify the Message, it returns false.

A composite signature allows two underlying signature algorithms to be combined into a single cryptographic signature operation and can be used for applications that require signatures.

2.3.1. Composite KeyGen

The KeyGen() -> (pk, sk) of a composite signature algorithm will perform the KeyGen() of the respective component signature algorithms and it produces a composite public key pk as per Section 3.2 and a composite secret key sk is per Section 3.3. The component keys MUST be uniquely generated for each component key of a Composite and MUST NOT be used in any other keys or as a standalone key.

2.3.2. Composite Sign

Generation of a composite signature involves applying each component algorithm's signature process to the input message according to its specification, and then placing each component signature value into the CompositeSignatureValue structure defined in Section 4.1.

The following process is used to generate composite signature values.

```
Sign (sk, Message) -> (signature)
Input:
     K1, K2              Signing private keys for each component. See note below o
n
                         composite inputs.

     A1, A2              Component signature algorithms. See note below on
                         composite inputs.

     Message             The Message to be signed, an octet string

     HASH                The Message Digest Algorithm used for pre-hashing.  See s
ection
                         on pre-hashing below.

     OID                 The Composite Signature String Algorithm Name converted
                         from ASCII to bytes.  See section on OID concatenation
                         below.

Output:
     signature           The composite signature, a CompositeSignatureValue

Signature Generation Process:

   1. Compute a Hash of the Message

         M' = HASH(Message)

   2. Generate the n component signatures independently,
      according to their algorithm specifications.

         S1 := Sign( K1, A1, DER(OID) || M' )
         S2 := Sign( K2, A2, DER(OID) || M' )

   3. Encode each component signature S1 and S2 into a BIT STRING
      according to its algorithm specification.

        signature ::= Sequence { S1, S2 }

   4. Output signature
```

Figure 1: Composite Sign(sk, Message)

Note on composite inputs: the method of providing the list of component keys and algorithms is flexible and beyond the scope of this pseudo-code. When passed to the Composite Sign(sk, Message) API the sk is a CompositePrivateKey. It is possible to construct a CompositePrivateKey from component keys stored in separate software or hardware keystores. Variations in the process to accommodate particular private key storage mechanisms are considered to be conformant to this document so long as it produces the same output as the process sketched above.

Since recursive composite public keys are disallowed, no component signature may itself be a composite; ie the signature generation process MUST fail if one of the private keys K1 or K2 is a composite.

A composite signature MUST produce, and include in the output, a signature value for every component key in the corresponding CompositePublicKey, and they MUST be in the same order; ie in the output, S1 MUST correspond to K1, S2 to K2.

2.3.3. Composite Verify

Verification of a composite signature involves applying each component algorithm's verification process according to its specification.

Compliant applications MUST output "Valid signature" (true) if and only if all component signatures were successfully validated, and "Invalid signature" (false) otherwise.

The following process is used to perform this verification.

```
Composite Verify(pk, Message, signature)
Input:
     P1, P2              Public verification keys. See note below on
                         composite inputs.

     Message             Message whose signature is to be verified,
                         an octet string.

     signature           CompositeSignatureValue containing the component
                         signature values (S1 and S2) to be verified.

     A1, A2              Component signature algorithms. See note
                         below on composite inputs.

     HASH                The Message Digest Algorithm for pre-hashing.  See
                         section on pre-hashing the message below.
```

```
     OID               The Composite Signature String Algorithm Name converted
                       from ASCII to bytes.  See section on OID concatenation
                       below

Output:
    Validity (bool)    "Valid signature" (true) if the composite
                        signature is valid, "Invalid signature"
                        (false) otherwise.

Signature Verification Procedure::
   1. Check keys, signatures, and algorithms lists for consistency.

      If Error during Desequencing, or the sequences have
      different numbers of elements, or any of the public keys
      P1 or P2 and the algorithm identifiers A1 or A2 are
      composite then output "Invalid signature" and stop.

   2. Compute a Hash of the Message

         M' = HASH(Message)

   3. Check each component signature individually, according to its
       algorithm specification.
       If any fail, then the entire signature validation fails.

       if not verify( P1, DER(OID) || M', S1, A1 ) then
            output "Invalid signature"
       if not verify( P2, DER(OID) || M', S2, A2 ) then
            output "Invalid signature"

       if all succeeded, then
        output "Valid signature"
```

Figure 2: Composite Verify(pk, Message, signature)

Note on composite inputs: the method of providing the list of component keys and algorithms is flexible and beyond the scope of this pseudo-code. When passed to the Composite Verify(pk, Message, signature) API the pk is a CompositePublicKey. It is possible to construct a CompositePublicKey from component keys stored in separate software or hardware keystores. Variations in the process to accommodate particular private key storage mechanisms are considered to be conformant to this document so long as it produces the same output as the process sketched above.

Since recursive composite public keys are disallowed, no component signature may itself be a composite; ie the signature generation process MUST fail if one of the private keys K1 or K2 is a composite.

2.4. OID Concatenation

As mentioned above, the OID input value for the Composite Signature Generation and verification process is the DER encoding of the OID represented in Hexidecimal bytes. The following table shows the HEX encoding for each Signature AlgorithmID

Composite Signature AlgorithmID	DER Encoding to be prepended to each Message
id-MLDSA44-RSA2048-PSS-SHA256	060B6086480186FA6B50080101
id-MLDSA44-RSA2048-PKCS15-SHA256	060B6086480186FA6B50080102
id-MLDSA44-Ed25519-SHA512	060B6086480186FA6B50080103
id-MLDSA44-ECDSA-P256-SHA256	060B6086480186FA6B50080104
id-MLDSA44-ECDSA-brainpoolP256r1-SHA256	060B6086480186FA6B50080105
id-MLDSA65-RSA3072-PSS-SHA512	060B6086480186FA6B50080106
id-MLDSA65-RSA3072-PKCS15-SHA512	060B6086480186FA6B50080107
id-MLDSA65-ECDSA-P256-SHA512	060B6086480186FA6B50080108
id-MLDSA65-ECDSA-brainpoolP256r1-SHA512	060B6086480186FA6B50080109
id-MLDSA65-Ed25519-SHA512	060B6086480186FA6B5008010A
id-MLDSA87-ECDSA-P384-SHA512	060B6086480186FA6B5008010B
id-MLDSA87-ECDSA-brainpoolP384r1-SHA512	060B6086480186FA6B5008010C
id-MLDSA87-Ed448-SHA512	060B6086480186FA6B5008010D

Table 1: Composite Signature OID Concatenations

2.5. PreHashing the Message

As noted in the composite signature generation process and composite signature verification process, the Message should be pre-hashed into M' with the digest algorithm specified in the composite signature algorithm identifier. The choice of the digest algorithm was chosen with the following criteria:

1. For composites paired with RSA or ECDSA, the hashing algorithm SHA256 or SHA512 is used as part of the RSA or ECDSA signature algorithm and is therefore also used as the composite prehashing algorithm.

2. For ML-DSA signing a digest of the message is allowed as long as the hash function provides at least y bits of classical security strength against both collision and second preimage attacks. For MLDSA44 y is 128 bits, MLDSA65 y is 192 bits and for MLDSA87 y is 256 bits. Therefore SHA256 is paired with RSA and ECDSA with MLDSA44 and SHA512 is paired with RSA and ECDSA with MLDSA65 and MLDSA87 to match the appropriate security strength.

3. Ed25519 [RFC8032] uses SHA512 internally, therefore SHA512 is used to pre-hash the message when Ed25519 is a component algorithm.

4. Ed448 [RFC8032] uses SHAKE256 internally, but to reduce the set of prehashing algorihtms, SHA512 was selected to pre-hash the message when Ed448 is a component algorithm.

2.6. Algorithm Selection Criteria

The composite algorithm combinations defined in this document were chosen according to the following guidelines:

1. A single RSA combination is provided at a key size of 3072 bits, matched with NIST PQC Level 3 algorithms.

2. Elliptic curve algorithms are provided with combinations on each of the NIST [RFC6090], Brainpool [RFC5639], and Edwards [RFC7748] curves. NIST PQC Levels 1 - 3 algorithms are matched with 256-bit curves, while NIST levels 4 - 5 are matched with 384-bit elliptic curves. This provides a balance between matching classical security levels of post-quantum and traditional algorithms, and also selecting elliptic curves which already have wide adoption.

3. NIST level 1 candidates are provided, matched with 256-bit elliptic curves, intended for constrained use cases.

If other combinations are needed, a separate specification should be submitted to the IETF LAMPS working group. To ease implementation, these specifications are encouraged to follow the construction pattern of the algorithms specified in this document.

The composite structures defined in this specification allow only for pairs of algorithms. This also does not preclude future specification from extending these structures to define combinations with three or more components.

3. Composite Signature Structures

In order for signatures to be composed of multiple algorithms, we define encodings consisting of a sequence of signature primitives (aka "component algorithms") such that these structures can be used as a drop-in replacement for existing signature fields such as those found in PKCS#10 [RFC2986], CMP [RFC4210], X.509 [RFC5280], CMS [RFC5652].

3.1. pk-CompositeSignature

The following ASN.1 Information Object Class is a template to be used in defining all composite Signature public key types.

```
pk-CompositeSignature {OBJECT IDENTIFIER:id,
  FirstPublicKeyType,SecondPublicKeyType}
    PUBLIC-KEY ::= {
      IDENTIFIER id
      KEY SEQUENCE {
        firstPublicKey BIT STRING (CONTAINING FirstPublicKeyType),
        secondPublicKey BIT STRING (CONTAINING SecondPublicKeyType)
      }
      PARAMS ARE absent
      CERT-KEY-USAGE { digitalSignature, nonRepudiation, keyCertSign, cRLSign}
    }
```

As an example, the public key type pk-MLDSA65-ECDSA-P256-SHA256 is defined as:

```
pk-MLDSA65-ECDSA-P256-SHA256 PUBLIC-KEY ::=
  pk-CompositeSignature{ id-MLDSA65-ECDSA-P256-SHA256,
  OCTET STRING, ECPoint}
```

The full set of key types defined by this specification can be found in the ASN.1 Module in Section 6.

3.2. CompositeSignaturePublicKey

Composite public key data is represented by the following structure:

```
CompositeSignaturePublicKey ::= SEQUENCE SIZE (2) OF BIT STRING
```

A composite key MUST contain two component public keys. The order of the component keys is determined by the definition of the corresponding algorithm identifier as defined in section Section 5.

Some applications may need to reconstruct the SubjectPublicKeyInfo objects corresponding to each component public key. Table 3 in Section 5 provides the necessary mapping between composite and their component algorithms for doing this reconstruction. This also motivates the design choice of SEQUENCE OF BIT STRING instead of SEQUENCE OF OCTET STRING; using BIT STRING allows for easier transcription between CompositeSignaturePublicKey and SubjectPublicKeyInfo.

When the CompositeSignaturePublicKey must be provided in octet string or bit string format, the data structure is encoded as specified in Section 3.4.

Component keys of a CompositeSignaturePublicKey MUST NOT be used in any other type of key or as a standalone key.

3.3. CompositeSignaturePrivateKey

Usecases that require an interoperable encoding for composite private keys, such as when private keys are carried in PKCS #12 [RFC7292], CMP [RFC4210] or CRMF [RFC4211] MUST use the following structure.

CompositeSignaturePrivateKey ::= SEQUENCE SIZE (2) OF OneAsymmetricKey

Each element is a OneAsymmetricKey‘ [RFC5958] object for a component private key.

The parameters field MUST be absent.

The order of the component keys is the same as the order defined in Section 3.2 for the components of CompositeSignaturePublicKey.

When a CompositeSignaturePrivateKey is conveyed inside a OneAsymmetricKey structure (version 1 of which is also known as PrivateKeyInfo) [RFC5958], the privateKeyAlgorithm field SHALL be set to the corresponding composite algorithm identifier defined according to Section 5, the privateKey field SHALL contain the CompositeSignaturePrivateKey, and the publicKey field MUST NOT be present. Associated public key material MAY be present in the CompositeSignaturePrivateKey.

In some usecases the private keys that comprise a composite key may not be represented in a single structure or even be contained in a single cryptographic module; for example if one component is within

the FIPS boundary of a cryptographic module and the other is not; see {sec-fips} for more discussion. The establishment of correspondence between public keys in a CompositeSignaturePublicKey and private keys not represented in a single composite structure is beyond the scope of this document.

Component keys of a CompositeSignaturePrivateKey MUST NOT be used in any other type of key or as a standalone key.

3.4. Encoding Rules

Many protocol specifications will require that the composite public key and composite private key data structures be represented by an octet string or bit string.

When an octet string is required, the DER encoding of the composite data structure SHALL be used directly.

CompositeSignaturePublicKeyOs ::= OCTET STRING (CONTAINING CompositeSignatureP
ublicKey ENCODED BY der)

When a bit string is required, the octets of the DER encoded composite data structure SHALL be used as the bits of the bit string, with the most significant bit of the first octet becoming the first bit, and so on, ending with the least significant bit of the last octet becoming the last bit of the bit string.

CompositeSignaturePublicKeyBs ::= BIT STRING (CONTAINING CompositeSignaturePub
licKey ENCODED BY der)

In the interests of simplicity and avoiding compatibility issues, implementations that parse these structures MAY accept both BER and DER.

3.5. Key Usage Bits

For protocols such as X.509 [RFC5280] that specify key usage along with the public key, then the composite public key associated with a composite signature MUST have a signing-type key usage. This is because the composite public key can only be used in situations that are appropriate for both component algorithms, so even if the classical component key supports both signing and encryption, the post-quantum algorithms do not.

If the keyUsage extension is present in a Certification Authority (CA) certificate that indicates a composite key, then any combination of the following values MAY be present and any other values MUST NOT be present:

digitalSignature;
nonRepudiation;
keyCertSign; and
cRLSign.

If the keyUsage extension is present in an End Entity (EE) certificate that indicates a composite key, then any combination of the following values MAY be present and any other values MUST NOT be present:

digitalSignature; and
nonRepudiation;

4. Composite Signature Structures

4.1. sa-CompositeSignature

The ASN.1 algorithm object for a composite signature is:

```
sa-CompositeSignature {
  OBJECT IDENTIFIER:id,
    PUBLIC-KEY:publicKeyType }
    SIGNATURE-ALGORITHM ::= {
        IDENTIFIER id
        VALUE CompositeSignatureValue
        PARAMS ARE absent
        PUBLIC-KEYS { publicKeyType }
    }
```

The following is an explanation how SIGNATURE-ALGORITHM elements are used to create Composite Signatures:

SIGNATURE-ALGORITHM element	Definition
IDENTIFIER	The Object ID used to identify the composite Signature Algorithm
VALUE	The Sequence of BIT STRINGS for each component signature value
PARAMS	Parameters are absent
PUBLIC-KEYS	The composite key required to produce the composite signature

Table 2

4.2. CompositeSignatureValue

The output of the composite signature algorithm is the DER encoding of the following structure:

```
CompositeSignatureValue ::= SEQUENCE SIZE (2) OF BIT STRING
```

Where each BIT STRING within the SEQUENCE is a signature value produced by one of the component keys. It MUST contain one signature value produced by each component algorithm, and in the same order as specified in the object identifier.

The choice of SEQUENCE SIZE (2) OF BIT STRING, rather than for example a single BIT STRING containing the concatenated signature values, is to gracefully handle variable-length signature values by taking advantage of ASN.1's built-in length fields.

5. Algorithm Identifiers

This section defines the algorithm identifiers for explicit combinations. For simplicity and prototyping purposes, the signature algorithm object identifiers specified in this document are the same as the composite key object Identifiers. A proper implementation should not presume that the object ID of a composite key will be the same as its composite signature algorithm.

This section is not intended to be exhaustive and other authors may define other composite signature algorithms so long as they are compatible with the structures and processes defined in this and companion public and private key documents.

Some use-cases desire the flexibility for clients to use any combination of supported algorithms, while others desire the rigidity of explicitly-specified combinations of algorithms.

The following table summarizes the details for each explicit composite signature algorithms:

The OID referenced are TBD for prototyping only, and the following prefix is used for each:

replace <CompSig> with the String "2.16.840.1.114027.80.8.1"

Therefore <CompSig>.1 is equal to 2.16.840.1.114027.80.8.1.1

Signature public key types:

Composite Signature AlgorithmID	OID	First Algorithm	Second Algorithm	Pre-Hash
id-MLDSA44-RSA2048-PSS-SHA256	<CompSig>.1	MLDSA44	SHA256WithRSAPSS	SHA256
id-MLDSA44-RSA2048-PKCS15-SHA256	<CompSig>.2	MLDSA44	SHA256WithRSAEncryption	SHA256
id-MLDSA44-Ed25519-SHA512	<CompSig>.3	MLDSA44	Ed25519	SHA512
id-MLDSA44-ECDSA-P256-SHA256	<CompSig>.4	MLDSA44	SHA256withECDSA	SHA256
id-MLDSA44-ECDSA-brainpoolP256r1-SHA256	<CompSig>.5	MLDSA44	SHA256withECDSA	SHA256
id-MLDSA65-RSA3072-PSS-SHA512	<CompSig>.6	MLDSA65	SHA512WithRSAPSS	SHA512
id-MLDSA65-RSA3072-PKCS15-SHA512	<CompSig>.7	MLDSA65	SHA512WithRSAEncryption	SHA512
id-MLDSA65-ECDSA-P256-SHA512	<CompSig>.8	MLDSA65	SHA512withECDSA	SHA512
id-MLDSA65-ECDSA-brainpoolP256r1-SHA512	<CompSig>.9	MLDSA65	SHA512withECDSA	SHA512
id-MLDSA65-Ed25519-SHA512	<CompSig>.10	MLDSA65	Ed25519	SHA512
id-MLDSA87-ECDSA-P384-SHA512	<CompSig>.11	MLDSA87	SHA512withECDSA	SHA512

id-MLDSA87-ECDSA-brainpoolP384r1-SHA512	<CompSig>.12	MLDSA87	SHA512withECDSA	SHA512
id-MLDSA87-Ed448-SHA512	<CompSig>.13	MLDSA87	Ed448	SHA512

Table 3: Composite Signature Algorithms

The table above contains everything needed to implement the listed explicit composite algorithms. See the ASN.1 module in section Section 6 for the explicit definitions of the above Composite signature algorithms.

Full specifications for the referenced algorithms can be found as follows:

* _MLDSA_: [I-D.ietf-lamps-dilithium-certificates] and [FIPS.204-ipd]

* _ECDSA_: [RFC5480]

* _Ed25519 / Ed448_: [RFC8410]

* _RSAES-PKCS-v1_5_: [RFC8017]

* _RSASSA-PSS_: [RFC8017]

5.1. Notes on id-MLDSA44-RSA2048-PSS-SHA256

Use of RSA-PSS [RFC8017] deserves a special explanation.

The RSA component keys MUST be generated at the 2048-bit security level in order to match with ML-DSA-44

As with the other composite signature algorithms, when id-MLDSA44-RSA2048-PSS-SHA256 is used in an AlgorithmIdentifier, the parameters MUST be absent. id-MLDSA44-RSA2048-PSS-SHA256 SHALL instantiate RSA-PSS with the following parameters:

RSA-PSS Parameter	Value
Mask Generation Function	mgf1
Mask Generation params	SHA-256
Message Digest Algorithm	SHA-256

Table 4: RSA-PSS 2048 Parameters

where:

* Mask Generation Function (mgf1) is defined in [RFC8017]

* SHA-256 is defined in [RFC6234].

5.2. Notes on id-MLDSA65-RSA3072-PSS-SHA512

The RSA component keys MUST be generated at the 3072-bit security level in order to match with ML-DSA-65.

As with the other composite signature algorithms, when id-MLDSA65-RSA3072-PSS-SHA512 is used in an AlgorithmIdentifier, the parameters MUST be absent. id-MLDSA65-RSA3072-PSS-SHA512 SHALL instantiate RSA-PSS with the following parameters:

RSA-PSS Parameter	Value
Mask Generation Function	mgf1
Mask Generation params	SHA-512
Message Digest Algorithm	SHA-512

Table 5: RSA-PSS 3072 Parameters

where:

* Mask Generation Function (mgf1) is defined in [RFC8017]

* SHA-512 is defined in [RFC6234].

6. ASN.1 Module

<CODE STARTS>

```
   Composite-Signatures-2023
      { joint-iso-itu-t(2) country(16) us(840) organization(1) entrust(114027)
        algorithm(80) id-composite-signatures-2023 (TBDMOD) }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

EXPORTS ALL;

IMPORTS
  PUBLIC-KEY, SIGNATURE-ALGORITHM, AlgorithmIdentifier{}
    FROM AlgorithmInformation-2009  -- RFC 5912 [X509ASN1]
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-algorithmInformation-02(58) }

  SubjectPublicKeyInfo
    FROM PKIX1Explicit-2009
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-pkix1-explicit-02(51) }

  OneAsymmetricKey
    FROM AsymmetricKeyPackageModuleV1
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
        pkcs-9(9) smime(16) modules(0)
```

```
           id-mod-asymmetricKeyPkgV1(50) }

     RSAPublicKey, ECPoint
       FROM PKIXAlgs-2009
         { iso(1) identified-organization(3) dod(6)
           internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
           id-mod-pkix1-algorithms2008-02(56) }

     sa-rsaSSA-PSS
       FROM PKIX1-PSS-OAEP-Algorithms-2009
          {iso(1) identified-organization(3) dod(6) internet(1) security(5)
          mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-rsa-pkalgs-02(54)}

   ;

   --
   -- Object Identifiers
   --

   -- Defined in ITU-T X.690
   der OBJECT IDENTIFIER ::=
     {joint-iso-itu-t asn1(1) ber-derived(2) distinguished-encoding(1)}




   --
   -- Signature Algorithm
   --


   --
   -- Composite Signature basic structures
   --

   CompositeSignaturePublicKey ::= SEQUENCE SIZE (2) OF BIT STRING

   CompositeSignaturePublicKeyOs ::= OCTET STRING (CONTAINING
                                     CompositeSignaturePublicKey ENCODED BY der)

   CompositeSignaturePublicKeyBs ::= BIT STRING (CONTAINING
                                     CompositeSignaturePublicKey ENCODED BY der)

   CompositeSignaturePrivateKey ::= SEQUENCE SIZE (2) OF OneAsymmetricKey

   CompositeSignatureValue ::= SEQUENCE SIZE (2) OF BIT STRING

   -- Composite Signature Value is just a sequence of OCTET STRINGS
```

```
   --   CompositeSignaturePair{FirstSignatureValue, SecondSignatureValue} ::=
   --     SEQUENCE {
   --      signaturevalue1 FirstSignatureValue,
   --      signaturevalue2 SecondSignatureValue }

      -- An Explicit Compsite Signature is a set of Signatures which
      -- are composed of OCTET STRINGS
   --   ExplicitCompositeSignatureValue ::= CompositeSignaturePair {
   --       OCTET STRING,OCTET STRING}


   --
   -- Information Object Classes
   --

   pk-CompositeSignature {OBJECT IDENTIFIER:id,
     FirstPublicKeyType,SecondPublicKeyType}
       PUBLIC-KEY ::= {
         IDENTIFIER id
         KEY SEQUENCE {
           firstPublicKey BIT STRING (CONTAINING FirstPublicKeyType),
           secondPublicKey BIT STRING (CONTAINING SecondPublicKeyType)
         }
         PARAMS ARE absent
         CERT-KEY-USAGE { digitalSignature, nonRepudiation, keyCertSign, cRLSign}
       }


   sa-CompositeSignature{OBJECT IDENTIFIER:id,
      PUBLIC-KEY:publicKeyType }
         SIGNATURE-ALGORITHM ::=  {
            IDENTIFIER id
            VALUE CompositeSignatureValue
            PARAMS ARE absent
            PUBLIC-KEYS {publicKeyType}
         }

   -- TODO: OID to be replaced by IANA
   id-MLDSA44-RSA2048-PSS-SHA256 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 1 }

   pk-MLDSA44-RSA2048-PSS-SHA256 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA44-RSA2048-PSS-SHA256,
     OCTET STRING, RSAPublicKey}

   sa-MLDSA44-RSA2048-PSS-SHA256 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
```

```
        id-MLDSA44-RSA2048-PSS-SHA256,
        pk-MLDSA44-RSA2048-PSS-SHA256 }

   -- TODO: OID to be replaced by IANA
   id-MLDSA44-RSA2048-PKCS15-SHA256 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 2 }

   pk-MLDSA44-RSA2048-PKCS15-SHA256 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA44-RSA2048-PKCS15-SHA256,
     OCTET STRING, RSAPublicKey}

   sa-MLDSA44-RSA2048-PKCS15-SHA256 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA44-RSA2048-PKCS15-SHA256,
          pk-MLDSA44-RSA2048-PKCS15-SHA256 }


   -- TODO: OID to be replaced by IANA
   id-MLDSA44-Ed25519-SHA512 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 3 }

   pk-MLDSA44-Ed25519-SHA512 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA44-Ed25519-SHA512,
     OCTET STRING, ECPoint}

   sa-MLDSA44-Ed25519-SHA512 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA44-Ed25519-SHA512,
          pk-MLDSA44-Ed25519-SHA512 }


   -- TODO: OID to be replaced by IANA
   id-MLDSA44-ECDSA-P256-SHA256 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 4 }

   pk-MLDSA44-ECDSA-P256-SHA256 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA44-ECDSA-P256-SHA256,
     OCTET STRING, ECPoint}

   sa-MLDSA44-ECDSA-P256-SHA256 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA44-ECDSA-P256-SHA256,
          pk-MLDSA44-ECDSA-P256-SHA256 }
```

```
   -- TODO: OID to be replaced by IANA
   id-MLDSA44-ECDSA-brainpoolP256r1-SHA256 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 5 }

   pk-MLDSA44-ECDSA-brainpoolP256r1-SHA256 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA44-ECDSA-brainpoolP256r1-SHA256,
     OCTET STRING, ECPoint}

   sa-MLDSA44-ECDSA-brainpoolP256r1-SHA256 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA44-ECDSA-brainpoolP256r1-SHA256,
          pk-MLDSA44-ECDSA-brainpoolP256r1-SHA256 }


   -- TODO: OID to be replaced by IANA
   id-MLDSA65-RSA3072-PSS-SHA512 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 6 }

   pk-MLDSA65-RSA3072-PSS-SHA512 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA65-RSA3072-PSS-SHA512,
     OCTET STRING, RSAPublicKey}

   sa-MLDSA65-RSA3072-PSS-SHA512 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA65-RSA3072-PSS-SHA512,
          pk-MLDSA65-RSA3072-PSS-SHA512 }


   -- TODO: OID to be replaced by IANA
   id-MLDSA65-RSA3072-PKCS15-SHA512 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 7 }

   pk-MLDSA65-RSA3072-PKCS15-SHA512 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA65-RSA3072-PKCS15-SHA512,
     OCTET STRING, RSAPublicKey}

   sa-MLDSA65-RSA3072-PKCS15-SHA512 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA65-RSA3072-PKCS15-SHA512,
          pk-MLDSA65-RSA3072-PKCS15-SHA512 }


   -- TODO: OID to be replaced by IANA
   id-MLDSA65-ECDSA-P256-SHA512 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
```

```
      entrust(114027) algorithm(80) composite(8) signature(1) 8 }

   pk-MLDSA65-ECDSA-P256-SHA512 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA65-ECDSA-P256-SHA512,
     OCTET STRING, ECPoint}

   sa-MLDSA65-ECDSA-P256-SHA512 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA65-ECDSA-P256-SHA512,
          pk-MLDSA65-ECDSA-P256-SHA512 }


   -- TODO: OID to be replaced by IANA
   id-MLDSA65-ECDSA-brainpoolP256r1-SHA512 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 9 }

   pk-id-MLDSA65-ECDSA-brainpoolP256r1-SHA512 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA65-ECDSA-brainpoolP256r1-SHA512,
     OCTET STRING, ECPoint}

   sa-id-MLDSA65-ECDSA-brainpoolP256r1-SHA512 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA65-ECDSA-brainpoolP256r1-SHA512,
          pk-id-MLDSA65-ECDSA-brainpoolP256r1-SHA512 }


   -- TODO: OID to be replaced by IANA
   id-MLDSA65-Ed25519-SHA512 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 10 }

   pk-MLDSA65-Ed25519-SHA512 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA65-Ed25519-SHA512,
     OCTET STRING, ECPoint}

   sa-MLDSA65-Ed25519-SHA512 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA65-Ed25519-SHA512,
          pk-MLDSA65-Ed25519-SHA512 }


   -- TODO: OID to be replaced by IANA
   id-MLDSA87-ECDSA-P384-SHA512 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 11 }

   pk-MLDSA87-ECDSA-P384-SHA512 PUBLIC-KEY ::=
```

```
     pk-CompositeSignature{ id-MLDSA87-ECDSA-P384-SHA512,
     OCTET STRING, ECPoint}

   sa-MLDSA87-ECDSA-P384-SHA512 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA87-ECDSA-P384-SHA512,
          pk-MLDSA87-ECDSA-P384-SHA512 }


   -- TODO: OID to be replaced by IANA
   id-MLDSA87-ECDSA-brainpoolP384r1-SHA512 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 12 }

   pk-MLDSA87-ECDSA-brainpoolP384r1-SHA512 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA87-ECDSA-brainpoolP384r1-SHA512,
     OCTET STRING, ECPoint}

   sa-MLDSA87-ECDSA-brainpoolP384r1-SHA512 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA87-ECDSA-brainpoolP384r1-SHA512,
          pk-MLDSA87-ECDSA-brainpoolP384r1-SHA512 }


   -- TODO: OID to be replaced by IANA
   id-MLDSA87-Ed448-SHA512 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 13 }

   pk-MLDSA87-Ed448-SHA512 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-MLDSA87-Ed448-SHA512,
     OCTET STRING, ECPoint}

   sa-MLDSA87-Ed448-SHA512 SIGNATURE-ALGORITHM ::=
       sa-CompositeSignature{
          id-MLDSA87-Ed448-SHA512,
          pk-MLDSA87-Ed448-SHA512 }

   -- TODO: OID to be replaced by IANA
   id-Falon512-ECDSA-P256-SHA256 OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      entrust(114027) algorithm(80) composite(8) signature(1) 14 }

   pk-Falon512-ECDSA-P256-SHA256 PUBLIC-KEY ::=
     pk-CompositeSignature{ id-Falon512-ECDSA-P256-SHA256,
     OCTET STRING, ECPoint}

   sa-Falon512-ECDSA-P256-SHA256 SIGNATURE-ALGORITHM ::=
```

```
      sa-CompositeSignature{
         id-Falon512-ECDSA-P256-SHA256,
         pk-Falon512-ECDSA-P256-SHA256 }


   END
```

<CODE ENDS>

7. IANA Considerations

IANA is requested to allocate a value from the "SMI Security for PKIX Module Identifier" registry [RFC7299] for the included ASN.1 module, and allocate values from "SMI Security for PKIX Algorithms" to identify the fourteen Algorithms defined within.

7.1. Object Identifier Allocations

EDNOTE to IANA: OIDs will need to be replaced in both the ASN.1 module and in Table 3.

7.1.1. Module Registration - SMI Security for PKIX Module Identifier

* Decimal: IANA Assigned - *Replace TBDMOD*
* Description: Composite-Signatures-2023 - id-mod-composite-signatures
* References: This Document

7.1.2. Object Identifier Registrations - SMI Security for PKIX Algorithms

* id-MLDSA44-RSA2048-PSS-SHA256
* Decimal: IANA Assigned
* Description: id-MLDSA44-RSA2048-PSS-SHA256
* References: This Document
* id-MLDSA44-RSA2048-PKCS15-SHA256
* Decimal: IANA Assigned
* Description: id-MLDSA44-RSA2048-PKCS15-SHA256
* References: This Document

* id-MLDSA44-Ed25519-SHA512
* Decimal: IANA Assigned
* Description: id-MLDSA44-Ed25519-SHA512
* References: This Document
* id-MLDSA44-ECDSA-P256-SHA256
* Decimal: IANA Assigned
* Description: id-MLDSA44-ECDSA-P256-SHA256
* References: This Document
* id-MLDSA44-ECDSA-brainpoolP256r1-SHA256
* Decimal: IANA Assigned
* Description: id-MLDSA44-ECDSA-brainpoolP256r1-SHA256
* References: This Document
* id-MLDSA65-RSA3072-PSS-SHA512
* Decimal: IANA Assigned
* Description: id-MLDSA65-RSA3072-PSS-SHA512
* References: This Document
* id-MLDSA65-RSA3072-PKCS15-SHA512
* Decimal: IANA Assigned
* Description: id-MLDSA65-RSA3072-PKCS15-SHA512
* References: This Document
* id-MLDSA65-ECDSA-P256-SHA512
* Decimal: IANA Assigned
* Description: id-MLDSA65-ECDSA-P256-SHA512
* References: This Document

* id-MLDSA65-ECDSA-brainpoolP256r1-SHA512
* Decimal: IANA Assigned
* Description: id-MLDSA65-ECDSA-brainpoolP256r1-SHA512
* References: This Document
* id-MLDSA65-Ed25519-SHA512
* Decimal: IANA Assigned
* Description: id-MLDSA65-Ed25519-SHA512
* References: This Document
* id-MLDSA87-ECDSA-P384-SHA512
* Decimal: IANA Assigned
* Description: id-MLDSA87-ECDSA-P384-SHA512
* References: This Document
* id-MLDSA87-ECDSA-brainpoolP384r1-SHA512
* Decimal: IANA Assigned
* Description: id-MLDSA87-ECDSA-brainpoolP384r1-SHA512
* References: This Document
* id-MLDSA87-Ed448-SHA512
* Decimal: IANA Assigned
* Description: id-MLDSA87-Ed448-SHA512
* References: This Document

8. Security Considerations

8.1. Policy for Deprecated and Acceptable Algorithms

Traditionally, a public key, certificate, or signature contains a single cryptographic algorithm. If and when an algorithm becomes deprecated (for example, RSA-512, or SHA1), then clients performing signatures or verifications should be updated to adhere to appropriate policies.

In the composite model this is less obvious since implementers may decide that certain cryptographic algorithms have complementary security properties and are acceptable in combination even though one or both algorithms are deprecated for individual use. As such, a single composite public key or certificate may contain a mixture of deprecated and non-deprecated algorithms.

Since composite algorithms are registered independently of their component algorithms, their deprecation can be handled indpendently from that of their component algorithms. For example a cryptographic policy might continue to allow id-MLDSA65-ECDSA-P256-SHA512 even after ECDSA-P256 is deprecated.

When considering stripping attacks, one need consider the case where an attacker has fully compromised one of the component algorithms to the point that they can produce forged signatures that appear valid under one of the component public keys, and thus fool a victim verifier into accepting a forged signature. The protection against this attack relies on the victim verifier trusting the pair of public keys as a single composite key, and not trusting the individual component keys by themselves.

Specifically, in order to achieve this non-separability property, this specification makes two assumptions about how the verifier will establish trust in a composite public key:

1. This specification assumes that all of the component keys within a composite key are freshly generated for the composite; ie a given public key MUST NOT appear as a component within a composite key and also within single-algorithm constructions.

2. This specification assumes that composite public keys will be bound in a structure that contains a signature over the public key (for example, an X.509 Certificate [RFC5280]), which is chained back to a trust anchor, and where that signature algorithm is at least as strong as the composite public key that it is protecting.

There are mechanisms within Internet PKI where trusted public keys do not appear within signed structures -- such as the Trust Anchor format defined in [RFC5914]. In such cases, it is the responsibility of implementers to ensure that trusted composite keys are distributed in a way that is tamper-resistant and does not allow the component keys to be trusted independently.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/info/rfc4211>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

[RFC5639] Lochter, M. and J. Merkle, "Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation", RFC 5639, DOI 10.17487/RFC5639, March 2010, <https://www.rfc-editor.org/info/rfc5639>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/info/rfc5958>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, DOI 10.17487/RFC6090, February 2011, <https://www.rfc-editor.org/info/rfc6090>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8410] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure", RFC 8410, DOI 10.17487/RFC8410, August 2018, <https://www.rfc-editor.org/info/rfc8410>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/info/rfc8411>.

[X.690] ITU-T, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ISO/IEC 8825-1:2015, November 2015.

9.2. Informative References

[ANSSI2024]
French Cybersecurity Agency (ANSSI), Federal Office for Information Security (BSI), Netherlands National Communications Security Agency (NLNCSA), and Swedish National Communications Security Authority, Swedish Armed Forces, "Position Paper on Quantum Key Distribution", n.d., <https://cyber.gouv.fr/sites/default/files/document/Quantum_Key_Distribution_Position_Paper.pdf>.

[Bindel2017]
Bindel, N., Herath, U., McKague, M., and D. Stebila, "Transitioning to a quantum-resistant public key infrastructure", 2017, <https://link.springer.com/chapter/10.1007/978-3-319-59879-6_22>.

[BSI2021] Federal Office for Information Security (BSI), "Quantum-safe cryptography - fundamentals, current developments and recommendations", October 2021, <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.pdf>.

[I-D.becker-guthrie-noncomposite-hybrid-auth]
Becker, A., Guthrie, R., and M. J. Jenkins, "Non-Composite Hybrid Authentication in PKIX and Applications to Internet Protocols", Work in Progress, Internet-Draft, draft-becker-guthrie-noncomposite-hybrid-auth-00, 22 March 2022, <https://datatracker.ietf.org/doc/html/draft-becker-guthrie-noncomposite-hybrid-auth-00>.

[I-D.driscoll-pqt-hybrid-terminology]
D, F., "Terminology for Post-Quantum Traditional Hybrid Schemes", Work in Progress, Internet-Draft, draft-driscoll-pqt-hybrid-terminology-01, 20 October 2022, <https://datatracker.ietf.org/doc/html/draft-driscoll-pqt-hybrid-terminology-01>.

[I-D.guthrie-ipsecme-ikev2-hybrid-auth]
Guthrie, R., "Hybrid Non-Composite Authentication in IKEv2", Work in Progress, Internet-Draft, draft-guthrie-ipsecme-ikev2-hybrid-auth-00, 25 March 2022, <https://datatracker.ietf.org/doc/html/draft-guthrie-ipsecme-ikev2-hybrid-auth-00>.

[I-D.hale-pquip-hybrid-signature-spectrums]
Bindel, N., Hale, B., Connolly, D., and F. D, "Hybrid signature spectrums", Work in Progress, Internet-Draft, draft-hale-pquip-hybrid-signature-spectrums-01, 6 November 2023, <https://datatracker.ietf.org/doc/html/draft-hale-pquip-hybrid-signature-spectrums-01>.

[I-D.ietf-lamps-dilithium-certificates]
Massimo, J., Kampanakis, P., Turner, S., and B. Westerbaan, "Internet X.509 Public Key Infrastructure: Algorithm Identifiers for Dilithium", Work in Progress, Internet-Draft, draft-ietf-lamps-dilithium-certificates-01, 6 February 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-dilithium-certificates-01>.

[I-D.massimo-lamps-pq-sig-certificates]
Massimo, J., Kampanakis, P., Turner, S., and B. Westerbaan, "Algorithms and Identifiers for Post-Quantum Algorithms", Work in Progress, Internet-Draft, draft-massimo-lamps-pq-sig-certificates-00, 8 July 2022, <https://datatracker.ietf.org/doc/html/draft-massimo-lamps-pq-sig-certificates-00>.

[I-D.ounsworth-pq-composite-kem]
Ounsworth, M. and J. Gray, "Composite KEM For Use In Internet PKI", Work in Progress, Internet-Draft, draft-ounsworth-pq-composite-kem-01, 13 March 2023, <https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-kem-01>.

[I-D.pala-klaussner-composite-kofn]
Pala, M. and J. Klaußner, "K-threshold Composite Signatures for the Internet PKI", Work in Progress, Internet-Draft, draft-pala-klaussner-composite-kofn-00, 15 November 2022, <https://datatracker.ietf.org/doc/html/draft-pala-klaussner-composite-kofn-00>.

[I-D.vaira-pquip-pqc-use-cases]
Vaira, A., Brockhaus, H., Railean, A., Gray, J., and M. Ounsworth, "Post-quantum cryptography use cases", Work in Progress, Internet-Draft, draft-vaira-pquip-pqc-use-cases-00, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-vaira-pquip-pqc-use-cases-00>.

[RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, DOI 10.17487/RFC3279, April 2002, <https://www.rfc-editor.org/info/rfc3279>.

[RFC7292] Moriarty, K., Ed., Nystrom, M., Parkinson, S., Rusch, A., and M. Scott, "PKCS #12: Personal Information Exchange Syntax v1.1", RFC 7292, DOI 10.17487/RFC7292, July 2014, <https://www.rfc-editor.org/info/rfc7292>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[RFC7299] Housley, R., "Object Identifier Registry for the PKIX Working Group", RFC 7299, DOI 10.17487/RFC7299, July 2014, <https://www.rfc-editor.org/info/rfc7299>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

Appendix A. Samples

A.1. Explicit Composite Signature Examples

A.1.1. MLDSA44-ECDSA-P256-SHA256 Public Key

```
-----BEGIN PUBLIC KEY-----
MIIFgTANBgtghkgBhvprUAgBBAOCBW4AMIIFaQOCBSEAJaSzbEOXCT27FgXshv87
2HLTgePmYCJCH2OVUi/PB9YTyBXXnw+smoXT4w0pcq3WPs7qQXz6GKj7R0mFfTjp
Rd6uH3hgdS5cbg+PwMWsRKigE6mWFpMwrliS8CfR2yYgjhRav7wGa4ja7RdmZoLz
T8UBN2Yg6P/KceWA1gX6rdVUalrUvmcfR64ry06IfotXXNFwQc3vI6s7khHSUZX5
Rsw55RK3E0ElNpZxfFHv17d2xwFkGRAYqJao+qo37WtfG6Ynx4cqQyLJzlRn++5R
G6K1nCwqhErpk4vDR2uHIwAPiW0StX9ZbBjO2smRTIuWS2WhmhZwJkDqSHmCiRI2
tPsxCtLpM8t2IhTVy/ObAdQGPDngTNIPH8kuoRrBhWGIiWJMlo8LkImCRt5m/8Di
```

```
aL8C2BQNL+BWBBcak/JZrLkKZOZM7pFwWruHVEd0608XerfiVO3ypqAxImJ2xcdD
kLys4jDlEMsC3oz4RQGXahj2Pr8Jxu8i0TIDDdV5MZw9wId/m+0/vSD8BOAu09Wu
V6ppUWkDZLHlzf12zx3ZzBF/CMqZNsxMdTFNbu2qQ2/CZMlEvZ9f0gxn6qNf8NHC
UqdeRr7p9z8PuGHErLHqCvQMrzia71cD4URV//SR8EUQkoo9imtw3XT2uKGUIjT/
dDyqWl8BlAZ64dUp9EWmHwG1cyKBcu2dtD0d4BMol1g4TOF7u/3hHcgOoiR+ON/3
7MoxkX9mHt6tP7hkVWy0Mb3Sjej12DG75D9z8gAzHyQhOs3suNliCzCUmUVYm5Mv
WdySiuShm6yu+9Ah+GqvESuNr/h6s1gZGbdCe9llGdFPniilhL5J9oDgYMp+wi2I
EeOugOFoaY1e2OI1OPjBpg1071ko9B3CGD+0PkPvKYmMGs0HTnzFWCLPj5dG2kg7
PEltarKvLVTIxrbjw03l3SXmqpNPU8SqFJ7hB3OpFJgjqL95IRTa68UM8aaUKLMa
Qjjx08+e13P3wwY3niCR5U751fus4ArGLN2JgfPB7bPSdz043PIvxsCYZxUQXSW3
xWhWQqaHJLml19obvnf/tEXQZLheAr7hOEb/UTNUIBj/A6LfB0Gs012B1aXfne4W
9K/OFXc9p0C7aWIfjfMrA/idOrd1Eoo2NGLid+wp8aXyDZkCf5OUretEFHqQcQ2J
znh8R4mh2Tf7hT8+Gj5Su6bZggHi9iIJZ1G7i0j4Wm3g6DJAXF6KbChMayKRunDp
k6Nm5iOeTmT+Vi4OJncuI6HezZMzO2s+2iY33uDL7tFR8fVn7dQiF78c1aNhWjfm
fIsLNQdZxt6orvnwSrZpdVhOtAu+vYVaEAShdHgfzvPSDHIjgyxs6mGdk0uDsGpP
f5d3e9KV40rXir2OXaYMOq2KTkLb6KHHxZayLG0D9/qSBOnSE/aXNhh1cHtKeYAe
jjXmfzsmgNELPNxFRrx8pEHG1Se0GJNJVZE9u6B2r9f09TgTxgPX/6XpBNUrlz21
fsIvNpRL48cwLHOCgYP/SAgE3gzRC6G5NEE19wQZHsFNGeUeGvrvUQgTyT1YwLx+
Abvp57bVjgLWli185/K1a8BmJ204RHfDhSFe7sVAIoI2pUcz7ydb178DCAvupP20
CxUIkgOk3C+cgUzTwsFU4iiix282ZBa8/nTUnH9r3IDJQJwdWtMCnByCc43UeVSh
WV3isRF+ANl6lSevNj0uzGE07a5gPahctBWMmevh6qFcv5XucwNRe7en96o7CgK+
5QNCAAReie6V/SXhsV0+AAEPt/7UjJqzbrZU1ZHKBLCDbX1cv1Zkpy+SabE2Pfpd
K7SzfBpZw0txE+bjIUT4j3zjgIDa -----END PUBLIC KEY-----
```

A.1.2. MLDSA44-ECDSA-P256 Private Key

```
-----BEGIN PRIVATE KEY----- MIIP6gIBADANBgtghkgBhvprUAgBBASCD9Qwgg/
QMIIPNgIBADANBgsrBgEEAQKC CwwEBASCDyAlpLNsQ5cJPbsWBeyG/
zvYctOB4+ZgIkIfY5VSL88H1oyY3xD+KvF2
4bnFPT7nM4+8rPjsCuuk2PyUr7vzEHf++3ovDEFUsHo88ez4zfzBeW+Aho6yC4Gf
H8BFsgVYv0HmRUPUEmVhrJUKt8VwHbOVyak7a4JWl6MWpwxnRYqso1rCcWKWZKQC
bZIQQhGIKaIyZQSVRSI4bdBAShuBaQPGkBwUDgM3QQgGbhHDSBqzLQk4MooWSUk0
Zhk4JCAgUhg0ZVkkECOYTYBIMRQGjFICSaMiIYAAShMTJVAwbaFCakg0QcFCZolC
CmBEUQm5cQESASEQDCSgRGAyhMQGDWOAKRkZZESiAOI0KWAWbaIkbBoUJWOiiQk5
hhBCadqGhZEYglhIQACkMJo4kmEwSIDGBYtELcCoAVuGJMPAhCOFDRE1RCSGLAq0
AAA1Cgk3CgjAZYEUKovCaBs5hti2CZO0DePECBvHZASnMdmmURsxRFA4YRxGUOSy
EeEyUIA2isM0LRwJAgEAjGHGLRySJYjAgROBYQO1UUSCSRjACdg0ZAGgABQHjAkm
juOUQBpEgNy0jOPGIAzETdyIhZMIMVRCZgyyRUAgLQQwcchIbFIWRswgIQHAJQMV
RJgQJFqERFuiJQqjBIsChVpEMEsCKFqwCFgkciRHIcwCICKUAEgWScTESdgiYWQy
MtCgIISkARMZRAg3ieMGKlBITMM2iosCgiQ2MIKiKeI2TFhGaQKWAckEJmE4hhII
bSEyjQuECMlASBTBJFlIahIHaRhAAeQoMRQSDhQCcsiwYYg4BiETCAwiMRwwEcyU
jAoBDgLDDJmCjZOwRECIkVQ0bdoiKgwmUlSoSFsASsOoIQNHAdSwLFG4aSMIChlB
YBmmUEK0RIA4LiEmIAREJJEkIlpCJAnJDBS5bZQ2QAI2juHIKMkyihOnARlDiSQG
KEJAkgijEQwZRkgCJqFIiqIWRRo5JNqCTQoXhNIihko2YCICRQvIhQuALVmIhUAQ
TlRCasFAJJAQISG0ZQgTYgk0bEIiKZG0BYikBSEUYovCCBgZCgTJJNwoSFwIaMEC
JBm0cUQQSMpALAKwDKEkZSOCIFySQWAAYAGYLcw4IVPIMQiYaBoijduAMVrAjCQl
hdowJhpHKCCIKEMmbeEghRKiRRmgZAHAZRRDMclGICGEZKO0ReM2EFgwFoabywZ8
xF2nuUHyMuPJG+EpYRxQCAJaxByDrIGke83F1KvEG5cCOPOXhi2O7cC4sxSopILK
```

```
I5bRDIFdHk15n9mCLq6w/K72nQSX4usU4izDewR/JZQZV6AR9D/yBSNyAfr50Z7U
pMrfq7NZp+KstrHxs0SBDm5XNBXBaIpAwIHKkL/epe+2q2FI4Dcy1rZAQMVX80UI
SmhaL1CRud4Yi1pEnPf0taOiaE8hjgQwoZ1DUl6HviFsJwaa2+y3RAjsGZuIVRTC
3nirZrYPw+dWVFWvMqvjh97SYgC7eTn2R4Xu3f/2GVaiaXgCEBhHTY5Ar8Fz6E3L
MVi9hWMMF6Mj5vXcdUhwA2aeXGjnqPdtA/jmZUXQlK3NmxS4v2ga0z7wFRiv2nNF
bnEGP6T/8oTcd78NrWobcQHfV3uWbbSwOhPrNgBWzuMl/FG9nMLRcQ8OZzjwxPgV
PGnW0tm/g2iAG30HtuagQP7OTpibqBurtEE+liVSFgoD0or4zYhPk3mz+OMVm5/4
1DdLLTRtnq7XVTVziy2JpTJVBeF8VDh5G9gQj8iFhXCNfj5ZFNfmjaHCE2oVefc4
FYVEr4MuoPq0bFmsbAlZMJYJ3d/zvoIx0Wj5IMGUrFNTl2fWzQyc75QMRcdKi2e0
YUcsFd6aWaHx/dhSid5PwXNryS8hsB4Uf2o2hvn5VXdUD7b9ZhDHvZy9l/RaRZWr
ih6pwSQ6HE0o6RcZJaAl2rG5i/TCkQN8Z+hOl6rxVSS0UesvhntrVEgo8hXVTnSi
3FRfWG/AWvDPe/GNRiDCRBBS2S2cNvJrNtpgzxXUh+8oPrOTylDQbn1fXd9wCCey
bdUj4cHvYJra/XmXOUws+qcUF5wa7aXRm8suk/nrvVws+lOU48C4fvsit1rQbbqG
0SJsasIpu4x0020zpxFm10TO619yAIeBY0VYDr28dckjJlr5latUFgZY5LcS4IHl
pxgZa6lA/UcAxgnnc04PWL58ErsZjGH2UL5TF2B1AVLRXVk1g4toM08vMpCWMucy
nvHIPfxBNT9Io2sVfnf5HL6OjCVaFM0evKQ00lyrRx/ASfkjJKl0Vz8da01ZmaCF
jUfB2pMVvqJvogflrKwOzjsPbAQbBdTG54dd2vSdCSdxHTcHf6MyxS0xrPIUBRjR
irUPU2ZzbTa/M5qfSMEsv6GMLDjtbn00ckBhq7Xc8ecHXmhPWP4PIC36deXorFVS
4sEcWbXQmNTJyYG0ex3Id9e+KZyKGwQm0Ts49sH8GCEd4VNJ48WdI/YrNBmz67RQ
HN0GIxOufDGiFxFrWjN7gOk0V75G2sLv6piA4N+2suvHYVyl0E7FGhUzTtDxhnAi
VAGNqFaYgp/UnpbLy6FuRQFop/6QT5nKB0v0LSwwdjKCFPXbCK34ybuTgV5RHCQJ
58CIyxcciGn7j0CIOZdT9F4zW4zIdjPvEcGQVWQtseknxA8DUoY9y0XIqDhFRd0y
1Q/4ynNFriItA83Q9XJk+DBrErRaXAcAc1Ti5jkNznyR5IGiHaliPIVhrYisVicm
+J1W93tf/4PgxEmB3aRPQPQK5xysT3Vh74bwXfIKRwQ4VLFPoSWuMh3hVtHB1dr6
NY3jELf5DXg2GYJ9cnfOiGhCNmVBb9KI/aCWPxg45wZpwI3DtS7ueia/qBDY5N+0
UDDdUkW6EP1pkdvhBHyhAHTUdS6P/V00O0PolSWwCNwXP/x5xmiz6yvy3YHg9b8H
4e+5HP3Fc6Q6jTRnZSB8BJVcLbhMIKDf3KP1iGdz7wKfAfgl/lJWkm+N9lv8196U
9208fbv6HttjU3zOxH2oOVaptDOGdA024TciOR/FS/nL+q4zh8lDcRFo4sKPUNQt
wYwDTJGMdxQ0aGhOIOEaDJ70HHvhq74hwwfn7DSimqGJvEub2kKkDPjuQwv/+P1R
4jV18QffWlQZMVa+OdOSYU2pI05XfGCapCHv3ZoD/orpNCEV0auMa1vKdQObOBOS
/6TIcVojw3tHKqtqSDYUjihFEeFYq3vLrfe9C/DAapim9M5P8ZlyndCJrXgr2Kg0
RnkoXJ7kTGK7cyzdudhaS+b5JaJOf3tsC64XMh3o0s9OdMwKqhbLIpGuWYOeiqm9
pRdaSPpc+NtzWTLIyVPlm22mGwpnjkkilwpFj3qtd1G1yYlGThn4/BvcOAA+qCH2
2+yeL1MzdTXVVTYy/nbux7WwZ5RXJmJOJaSzbEOXCT27FgXshv872HLTgePmYCJC
H2OVUi/PB9YTyBXXnw+smoXT4w0pcq3WPs7qQXz6GKj7R0mFfTjpRd6uH3hgdS5c
bg+PwMWsRKigE6mWFpMwrliS8CfR2yYgjhRav7wGa4ja7RdmZoLzT8UBN2Yg6P/K
ceWA1gX6rdVUalrUvmcfR64ry06IfotXXNFwQc3vI6s7khHSUZX5Rsw55RK3E0El
NpZxfFHv17d2xwFkGRAYqJao+qo37WtfG6Ynx4cqQyLJzlRn++5RG6K1nCwqhErp
k4vDR2uHIwAPiW0StX9ZbBjO2smRTIuWS2WhmhZwJkDqSHmCiRI2tPsxCtLpM8t2
IhTVy/ObAdQGPDngTNIPH8kuoRrBhWGIiWJMlo8LkImCRt5m/8DiaL8C2BQNL+BW
BBcak/JZrLkKZOZM7pFwWruHVEd0608XerfiVO3ypqAxImJ2xcdDkLys4jDlEMsC
3oz4RQGXahj2Pr8Jxu8i0TIDDdV5MZw9wId/m+0/vSD8BOAu09WuV6ppUWkDZLHl
zf12zx3ZzBF/CMqZNsxMdTFNbu2qQ2/CZMlEvZ9f0gxn6qNf8NHCUqdeRr7p9z8P
uGHErLHqCvQMrzia71cD4URV//SR8EUQkoo9imtw3XT2uKGUIjT/dDyqWl8BlAZ6
4dUp9EWmHwG1cyKBcu2dtD0d4BMol1g4TOF7u/3hHcgOoiR+ON/37MoxkX9mHt6t
P7hkVWy0Mb3Sjej12DG75D9z8gAzHyQhOs3suNliCzCUmUVYm5MvWdySiuShm6yu
+9Ah+GqvESuNr/h6s1gZGbdCe9llGdFPniilhL5J9oDgYMp+wi2IEeOugOFoaY1e
2OI1OPjBpg1071ko9B3CGD+0PkPvKYmMGs0HTnzFWCLPj5dG2kg7PEltarKvLVTI
```

```
xrbjw03l3SXmqpNPU8SqFJ7hB3OpFJgjqL95IRTa68UM8aaUKLMaQjjx08+e13P3
wwY3niCR5U751fus4ArGLN2JgfPB7bPSdz043PIvxsCYZxUQXSW3xWhWQqaHJLml
19obvnf/tEXQZLheAr7hOEb/UTNUIBj/A6LfB0Gs012B1aXfne4W9K/OFXc9p0C7
aWIfjfMrA/idOrd1Eoo2NGLid+wp8aXyDZkCf5OUretEFHqQcQ2Jznh8R4mh2Tf7
hT8+Gj5Su6bZggHi9iIJZ1G7i0j4Wm3g6DJAXF6KbChMayKRunDpk6Nm5iOeTmT+
Vi4OJncuI6HezZMzO2s+2iY33uDL7tFR8fVn7dQiF78c1aNhWjfmfIsLNQdZxt6o
rvnwSrZpdVhOtAu+vYVaEAShdHgfzvPSDHIjgyxs6mGdk0uDsGpPf5d3e9KV40rX
ir2OXaYMOq2KTkLb6KHHxZayLG0D9/qSBOnSE/aXNhh1cHtKeYAejjXmfzsmgNEL
PNxFRrx8pEHG1Se0GJNJVZE9u6B2r9f09TgTxgPX/6XpBNUrlz21fsIvNpRL48cw
LHOCgYP/SAgE3gzRC6G5NEE19wQZHsFNGeUeGvrvUQgTyT1YwLx+Abvp57bVjgLW
li185/K1a8BmJ204RHfDhSFe7sVAIoI2pUcz7ydb178DCAvupP20CxUIkgOk3C+c
gUzTwsFU4iiix282ZBa8/nTUnH9r3IDJQJwdWtMCnByCc43UeVShWV3isRF+ANl6
lSevNj0uzGE07a5gPahctBWMmevh6qFcv5XucwNRe7en96o7CgK+5TCBkwIBADAT
BgcqhkjOPQIBBggqhkjOPQMBBwR5MHcCAQEEIGS03QxzabfR/cKWFDEkvue30guK
RrdY2Lm6isBSMZfZoAoGCCqGSM49AwEHoUQDQgAEXonulf0l4bFdPgABD7f+1Iya
s262VNWRygSwg219XL9WZKcvkmmxNj36XSu0s3waWcNLcRPm4yFE+I9844CA2g==
-----END PRIVATE KEY-----
```

A.1.3. MLDSA44-ECDSA-P256 Self-Signed X509 Certificate

```
-----BEGIN CERTIFICATE-----
MIIP+TCCBhqgAwIBAgIUEsa5EwG0Ligbb3NMHEmsqr0IoKMwDQYLYIZIAYb6a1AI
AQQwEjEQMA4GA1UEAwwHb3FzdGVzdDAeFw0yNDAzMDEyMzE5MzFaFw0yNTAzMDEy
MzE5MzFaMBIxEDAOBgNVBAMMB29xc3Rlc3QwggWBMA0GC2CGSAGG+mtQCAEEA4IF
bgAwggVpA4IFIQAlpLNsQ5cJPbsWBeyG/zvYctOB4+ZgIkIfY5VSL88H1hPIFdef
D6yahdPjDSlyrdY+zupBfPoYqPtHSYV9OOlF3q4feGB1LlxuD4/AxaxEqKATqZYW
kzCuWJLwJ9HbJiCOFFq/vAZriNrtF2ZmgvNPxQE3ZiDo/8px5YDWBfqt1VRqWtS+
Zx9HrivLToh+i1dc0XBBze8jqzuSEdJRlflGzDnlErcTQSU2lnF8Ue/Xt3bHAWQZ
EBiolqj6qjfta18bpifHhypDIsnOVGf77lEborWcLCqESumTi8NHa4cjAA+JbRK1
f1lsGM7ayZFMi5ZLZaGaFnAmQOpIeYKJEja0+zEK0ukzy3YiFNXL85sB1AY8OeBM
0g8fyS6hGsGFYYiJYkyWjwuQiYJG3mb/wOJovwLYFA0v4FYEFxqT8lmsuQpk5kzu
kXBau4dUR3TrTxd6t+JU7fKmoDEiYnbFx0OQvKziMOUQywLejPhFAZdqGPY+vwnG
7yLRMgMN1XkxnD3Ah3+b7T+9IPwE4C7T1a5XqmlRaQNkseXN/XbPHdnMEX8Iypk2
zEx1MU1u7apDb8JkyUS9n1/SDGfqo1/w0cJSp15Gvun3Pw+4YcSsseoK9AyvOJrv
VwPhRFX/9JHwRRCSij2Ka3DddPa4oZQiNP90PKpaXwGUBnrh1Sn0RaYfAbVzIoFy
7Z20PR3gEyiXWDhM4Xu7/eEdyA6iJH443/fsyjGRf2Ye3q0/uGRVbLQxvdKN6PXY
MbvkP3PyADMfJCE6zey42WILMJSZRVibky9Z3JKK5KGbrK770CH4aq8RK42v+Hqz
WBkZt0J72WUZ0U+eKKWEvkn2gOBgyn7CLYgR466A4WhpjV7Y4jU4+MGmDXTvWSj0
HcIYP7Q+Q+8piYwazQdOfMVYIs+Pl0baSDs8SW1qsq8tVMjGtuPDTeXdJeaqk09T
xKoUnuEHc6kUmCOov3khFNrrxQzxppQosxpCOPHTz57Xc/fDBjeeIJHlTvnV+6zg
CsYs3YmB88Hts9J3PTjc8i/GwJhnFRBdJbfFaFZCpockuaXX2hu+d/+0RdBkuF4C
vuE4Rv9RM1QgGP8Dot8HQazTXYHVpd+d7hb0r84Vdz2nQLtpYh+N8ysD+J06t3US
ijY0YuJ37CnxpfINmQJ/k5St60QUepBxDYnOeHxHiaHZN/uFPz4aPlK7ptmCAeL2
IglnUbuLSPhabeDoMkBcXopsKExrIpG6cOmTo2bmI55OZP5WLg4mdy4jod7NkzM7
az7aJjfe4Mvu0VHx9Wft1CIXvxzVo2FaN+Z8iws1B1nG3qiu+fBKtml1WE60C769
hVoQBKF0eB/O89IMciODLGzqYZ2TS4Owak9/l3d70pXjSteKvY5dpgw6rYpOQtvo
ocfFlrIsbQP3+pIE6dIT9pc2GHVwe0p5gB6ONeZ/OyaA0Qs83EVGvHykQcbVJ7QY
k0lVkT27oHav1/T1OBPGA9f/pekE1SuXPbV+wi82lEvjxzAsc4KBg/9ICATeDNEL
```

obk0QTX3BBkewU0Z5R4a+u9RCBPJPVjAvH4Bu+nnttWOAtaWLXzn8rVrwGYnbThE
d8OFIV7uxUAigjalRzPvJ1vXvwMIC+6k/bQLFQiSA6TcL5yBTNPCwVTiKKLHbzZk
Frz+dNScf2vcgMlAnB1a0wKcHIJzjdR5VKFZXeKxEX4A2XqVJ682PS7MYTTtrmA9
qFy0FYyZ6+HqoVy/le5zA1F7t6f3qjsKAr7lA0IABF6J7pX9JeGxXT4AAQ+3/tSM
mrNutlTVkcoEsINtfVy/VmSnL5JpsTY9+l0rtLN8GlnDS3ET5uMhRPiPfOOAgNqj
ITAfMB0GA1UdDgQWBBR72ZiceTDE3NvBPY4ryxmGCVY5pjANBgtghkgBhvprUAgB
BAOCCcgAMIIJwwOCCXUAQZt6/8sreFucKJFrqjKF21L5S9zuNhAA7Klvuqd0/4R3
3efTQ52tdQg345qqpW0BZRlgUDSWnq1Bmu3dyjcGDHY8LLvHyoOyTe02AMgwTW2c
UH6XpqeaGu1cab6l0owF2m5yGV9EmKYC8fky4JhEk2As83l2xlc4mVbA09NKfvQs
p94zzodQdCYFJ5VY62OubFsy3m6rlzaRCvJ+GLfpp1UeU0vxgqjIrPC3bDnId+gH
Wpkq+3lZLP/duGxYeXeeOm4zaQuaIUCJ0e/mFCFpbHywwgncay+0OQFDTFRYhNDY
ZlJR3AWLF47Grn/MhOEj5VO4GJvj+CjlGlOe3Fn6ABEjdRSDUOYykGg3NtnLIGkf
hM8j4hd+jfqGKwZrPNz/YaNC4YtKX1uwbjv1gGZSQ7I6zBPw37r1yblTAielXtPE
l1DuoFr3f05gzJJF6OuG6s7iBoUc5n1ovrn3UO85nCbSqqO2Ky15xsqTkFPnz4+g
JUOZ0SkqznulXIaMtJf/SmF4Pn23fmjqpOBTqOkBAOjloQIgSiQ90z45JU8Cy5sM
COU5d6shh9oM1BMIhFco9Zgxc/VF3WWd7ig9NT120HO6seAtMEMMSjtEUJsqWyQI
NnQvNVCTdG9joftmHnn5Uczar6HLr1HNYvjjXVudrp6ziKTFgblklGdcrqN0OTtx
hNSyacxucS5DWm5RGicRQaWb8Khob1lSn9nAZC0rOaChzjzJ0F5FYH2j3vL4ztLa
D5LMT+0FV5q0gkDHcowdQmWBiSwgyvJm+SMUmT/jyDPyf1VR71i1R3jeVqTu+kaw
ZkUpBztGzMLuPUrTlFeWa+QitEQegzObUw3zSRYaOpKa+Xz49zSLEEK5mOpKIvp0
qub+LXJXPBs38CUSxh2MRCOd8t5LlQCnKOQbVOu0UHvWmEhG/Oa/3j97PC1xrLOj
KTYSDrjhmwcPkJql5zzgVC+gBHcmeUWSutlp4HYskp0WyWc2fR4rKTawH6D3TCdu
IfBtlj80CWhNrWoaC+8yN+Y8rsogv0VoeUkVbneAg7J7TuJo9poTu/RFyEJ4AkIq
XzOxpjiwyUStVY5YH3x8zgquFau7bK8RQLowC8VFFWA8BFcgAgQkmasiipREpGaZ
PxFelJbl6en0UHS1VWDF0sdadypkY+hAcVGPEOt9pcWvP0o31ioybplBLe831d1q
TtD1TfBHXNr1cFErqfwryRUhV0JYNkwKGt6IcN1sfe+j23wo/b1BRDld4z1WL45o
7nebuv499egoHN5+AHFSGxleBVWvZe0iJzXWMqfLGfEkXg92qh6+vMxQW0Dsh0lD
mJP8xaD2cXZVN8+X1H7ESD7fC8MyKgtHAzuPcNjx0Zl8wihPNz/LBc9H1o9+LyDT
wcl4Mys3XDQqeg/efLB7W7eyWcCR/TeVO1sNmFIfMOlqbA04hqao3q9PdN4LlKCR
RjDqxGMCnE6+/YZSmWbB4CyYQsy4yXHss3SNKFdO4KccUUMVRTAcvSmHs1lUOIGI
OUaFJ6yw/jp5Wjybx7G5r6v/kvnlh5V3EfdW4srrbMtiSvDrNsc6/pV4+/WJ8lDa
T++u+phJpKAPbc2t0Tly/x6/O0L+PJLMlD9qK1nGYdttvEDsKgHe4jTkMytiYQp7
VolKPsbgLA5o5esMm4YVtfPOJNiRIPa6NlVgf/enNziReTZVgWi8g/yDdcdpMuS9
Op6Hbx0gdH7vfFNjAyuSorTzrUIlJPv5ZYdVw3qj+zcWybHAeozJfPQyXS5QoJm9
7mXfmNvtH620g5qG6C9XyoP7oIul17bMiX7A0/cg06y3pxWRVw/Tmn1sn4Er3jBg
XATF5+2R/euNmggiQAlYhXshKj401wQT+pFgFruB3/U2nSuZtTrOJXp33pcCEWjx
USmitfUl0znkW2dIM4lD7RUUftJKJnSDWYeJSyM3rijlOftc5BZontZuzzeNBxgs
HX7Q26p5SOJWbSjsffCTQfxxrxNGUxl8qVdDqtAhZjw+ZnfD8LiJhHcrYERTFxW7
MjjMV7tP06XAqybmayAWCtbgJlDbZb/thROfPmmaaymiCbnDKkQK0TLGNtiZ0ijy
yd5YLlc8c1lBpxsNe4Myvd4He743kKlS8Ep5YzrIqkwPde0fA6pnpHiUxoi4wmsK
NTDOlkV9Cly5tfrRkDyfNOqLMO0pqzDXRpATUpu8xey5s+GHDmhI1K1ojybN3MKh
TTYJ9Gla+52EhetT8a5Dm1q+5fFjpOPtDhJNKhdOOMZDRDCisf+SEeGi8Wjybw7n
x3w4hQZZpDzTknq3MZbmnlilphajEPYulLAgGAHmXEPHFMOT56cJxTCj5WUqksRh
/07PXQ62wVCpgZk7IUY0v7UF+dcomp8kHaLT1KGCD1v6XgD2uySyt9ha+3MEsOjk
PhfLWKeVAYFQX8maIajcLKckHE27rzDgJ66nhJU/YmxtvRo84EGfvkkqpDxVpMl1
KWVE5U6nXSNX9KAwVr4v0Gn+Lw52WMHZAhzWDGDUE8Pkb/a4l+Wbj8xDmDZXSNqx
RT9oAI2IN2yZV0nTY+tlmGYgOMrdNMr6KhtwE6E7Q6+84Cy4xKkwY3o4ob49SSCr
WHOp1mtvqUIkvfo9nV0qm3AlCd6blZGCUtYWJVhdLUY9+YqtqvAlZWom6TW9609n

xQ2pDbUnzz3RBSWotye/JZ3Iixsv1nlgf32+0x6IGutdL7SYQ8C4LiunoJTkF+A6
U5Chd9MiPvrUXjii1eiifwlwOZUxpvS50tBXIwQTuriV3AZPS5R15nJp1t4YzGOE
QwRJtNFnAkLsZ4iTrj/zDtgR6a2r4m/vzhUkrxIdZiptMO1E21zKfUMOtddPVqNg
SF4emCWJhmaoA0fQbRmZAQxqSajNFB4XPBK729A3wtTaN2TG1vm4iXzjsJuXhDzM
52NYEv8m2RoyKTiZFEfHndtvAHrqrdV1ud5jYi0GoA9YpnrdLiuzh1ljru3hk0WP
ejLjifxq6XEHkmBlsaiJSFpDxbhwwuW2zBvsUtbyU+Y8KxUuSKmem1m2JUE+UOIv
s7eJpbVOEA31p9Jd5rYlIPpuJ0gfRwyK65Rj20DYmdGoJpBPZbc2awktnniBMyI4
P1hehYqQl5+hqavvGjM3OExPVVuQl5vA2dzm6QYMECNOWV9jgsfV5/H3EhQiJTtx
dpigxtff/QAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA0dKzgDSAAwRQIgTgIe913r
u9h+Hh3v0dLD71pyH3EidqMddgc0vjd7qF8CIQCNyqGxNNM4DuRCkDTWH+HdmGJ1
F2ljUsBn8vo47P9JvA== -----END CERTIFICATE-----

Appendix B. Implementation Considerations

B.1. FIPS certification

One of the primary design goals of this specification is for the overall composite algorithm to be able to be considered FIPS-approved even when one of the component algorithms is not.

Implementors seeking FIPS certification of a composite Signature algorithm where only one of the component algorithms has been FIPS-validated or FIPS-approved should credit the FIPS-validated component algorithm with full security strength, the non-FIPS-validated component algorith with zero security, and the overall composite should be considered full strength and thus FIPS-approved.

The authors wish to note that this gives composite algorithms great future utility both for future cryptographic migrations as well as bridging across jurisdictions; for example defining composite algorithms which combine FIPS cryptography with cryptography from a different national standards body.

B.2. Backwards Compatibility

The term "backwards compatibility" is used here to mean something more specific; that existing systems as they are deployed today can interoperate with the upgraded systems of the future. This draft explicitly does not provide backwards compatibility, only upgraded systems will understand the OIDs defined in this document.

If backwards compatibility is required, then additional mechanisms will be needed. Migration and interoperability concerns need to be thought about in the context of various types of protocols that make use of X.509 and PKIX with relation to digital signature objects, from online negotiated protocols such as TLS 1.3 [RFC8446] and IKEv2 [RFC7296], to non-negotiated asynchronous protocols such as S/MIME signed email [RFC8551], document signing such as in the context of

the European eIDAS regulations [eIDAS2014], and publicly trusted code signing [codeSigningBRsv2.8], as well as myriad other standardized and proprietary protocols and applications that leverage CMS [RFC5652] signed structures. Composite simplifies the protocol design work because it can be implemented as a signature algorithm that fits into existing systems.

B.2.1. Parallel PKIs

We present the term "Parallel PKI" to refer to the setup where a PKI end entity possesses two or more distinct public keys or certificates for the same identity (name), but containing keys for different cryptographic algorithms. One could imagine a set of parallel PKIs where an existing PKI using legacy algorithms (RSA, ECC) is left operational during the post-quantum migration but is shadowed by one or more parallel PKIs using pure post quantum algorithms or composite algorithms (legacy and post-quantum).

Equipped with a set of parallel public keys in this way, a client would have the flexibility to choose which public key(s) or certificate(s) to use in a given signature operation.

For negotiated protocols, the client could choose which public key(s) or certificate(s) to use based on the negotiated algorithms, or could combine two of the public keys for example in a non-composite hybrid method such as [I-D.becker-guthrie-noncomposite-hybrid-auth] or [I-D.guthrie-ipsecme-ikev2-hybrid-auth]. Note that it is possible to use the signature algorithms defined in Section 5 as a way to carry the multiple signature values generated by one of the non-composite public mechanism in protocols where it is easier to support the composite signature algorithms than to implement such a mechanism in the protocol itself. There is also nothing precluding a composite public key from being one of the components used within a non-composite authentication operation; this may lead to greater convenience in setting up parallel PKI hierarchies that need to service a range of clients implementing different styles of post-quantum migration strategies.

For non-negotiated protocols, the details for obtaining backwards compatibility will vary by protocol, but for example in CMS [RFC5652], the inclusion of multiple SignerInfo objects is often already treated as an OR relationship, so including one for each of the signer’s parallel PKI public keys would, in many cases, have the desired effect of allowing the receiver to choose one they are compatible with and ignore the others, thus achieving full backwards compatibility.

B.2.2. Hybrid Extensions (Keys and Signatures)

The use of Composite Crypto provides the possibility to process multiple algorithms without changing the logic of applications, but updating the cryptographic libraries: one-time change across the whole system. However, when it is not possible to upgrade the crypto engines/libraries, it is possible to leverage X.509 extensions to encode the additional keys and signatures. When the custom extensions are not marked critical, although this approach provides the most backward-compatible approach where clients can simply ignore the post-quantum (or extra) keys and signatures, it also requires all applications to be updated for correctly processing multiple algorithms together.

Appendix C. Intellectual Property Considerations

The following IPR Disclosure relates to this draft:

https://datatracker.ietf.org/ipr/3588/

Appendix D. Contributors and Acknowledgements

This document incorporates contributions and comments from a large group of experts. The Editors would especially like to acknowledge the expertise and tireless dedication of the following people, who attended many long meetings and generated millions of bytes of electronic mail and VOIP traffic over the past few years in pursuit of this document:

Scott Fluhrer (Cisco Systems), Daniel Van Geest (ISARA), Britta Hale, Tim Hollebeek (Digicert), Panos Kampanakis (Cisco Systems), Richard Kisley (IBM), Serge Mister (Entrust), Francois Rousseau, Falko Strenzke, Felipe Ventura (Entrust) and Alexander Ralien (Siemens)

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

This document borrows text from similar documents, including those referenced below. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

D.1. Making contributions

Additional contributions to this draft are welcome. Please see the working copy of this draft at, as well as open issues at:

https://github.com/EntrustCorporation/draft-ounsworth-composite-sigs

Authors' Addresses

Mike Ounsworth
Entrust Limited
2500 Solandt Road -- Suite 100
Ottawa, Ontario K2K 3G5
Canada
Email: mike.ounsworth@entrust.com

John Gray
Entrust Limited
2500 Solandt Road -- Suite 100
Ottawa, Ontario K2K 3G5
Canada
Email: john.gray@entrust.com

Massimiliano Pala
OpenCA Labs
858 Coal Creek Circle
Louisville, Colorado, 80027
United States of America
Email: director@openca.org

Jan Klaussner
D-Trust GmbH
Kommandantenstr. 15
10969 Berlin
Germany
Email: jan.klaussner@d-trust.net